PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-168737

(43)Date of publication of application : 04.07.1995

(51)Int.Cl. G06F 11/28
G06F 9/46
G06F 11/30

(21)Application number : 06-235467

(22)Date of filing : 29.09.1994

(71)Applicant : PHILIPS ELECTRON NV

(72)Inventor : PAUL ANDRIEUX CLARK
JONATHAN RICHARD PIESING

(30)Priority

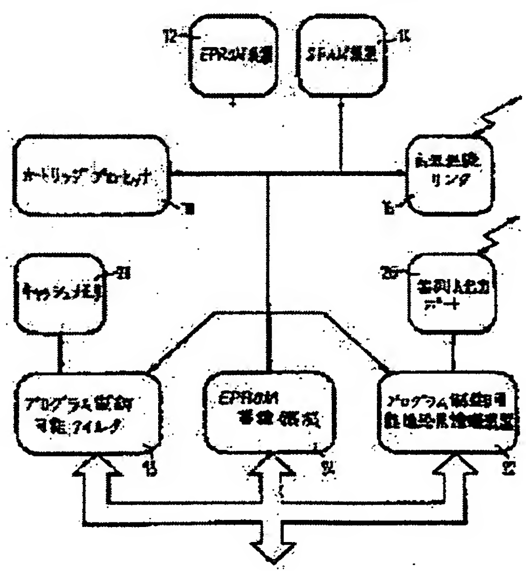
Priority number : 93 9320052 Priority date : 29.09.1993 Priority country : GB

(54) METHOD FOR MONITORING PROGRAM CONTROLLER

(57)Abstract:

PURPOSE: To unforcedly monitor the operation in real time without affecting the normal operation of a computer system.

CONSTITUTION: A monitoring device is provided with a program controllable event extraction filter 18 for discriminating the arrival of prescribed signals like memory address selection to the signal bus of the program controller of a CD player or the like. At the time of the detection of an event, two or three continuous passing signals are written through a buffer 20 to a storage device 14 and processed in an internal processor 10 or sent through a bidirectional communication link 16 to a host device and appropriate data processing algorithm is executed. The monitoring device is used for the unforced debugging of the operating system of the program controller.



CX

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平7－168737

(43)公開日　平成7年(1995)7月4日

(51)Int.Cl.⁶	識別記号	庁内整理番号	F I	技術表示箇所
G06F 11/28	A	9290－5B		
9/46	310 K	7629－5B		
11/30	A	9290－5B		

審査請求　未請求　請求項の数13　ＯＬ　（全 18 頁）

(21)出願番号　特願平6－235467

(22)出願日　平成6年(1994)9月29日

(31)優先権主張番号　９３２００５２：５

(32)優先日　1993年9月29日

(33)優先権主張国　イギリス（ＧＢ）

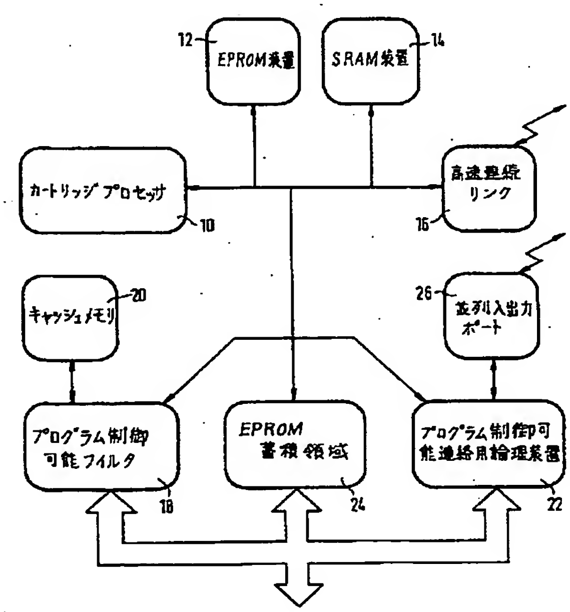
(71)出願人　592098322
フィリップス　エレクトロニクス　ネムローゼ　フェンノートシャップ
PHILIPS ELECTRONICS NEAMLOZE VENNOOTSHAP
オランダ国　5621　ベーアー　アインドーフェン　フルーネヴァウツウェッハ1

(74)代理人　弁理士　杉村　暁秀　（外5名）

最終頁に続く

(54)【発明の名称】　プログラム制御装置のモニタ方法

(57)【要約】
【目的】　コンピュータシステムの正常動作には影響を与えずにその動作を実時間で非強制的にモニタする。
【構成】　ＣＤプレーヤなどプログラム制御装置の信号バスにメモリ番地選定など所定信号の到来を判別するプログラム制御可能の事象抽出フィルタをモニタ装置に設け、事象の検出に際し、二，三の連続通過信号を、バッファ（20）を介し、記憶装置（14）に書込んで内部プロセサ（10）で処理し、もしくは双方向通信リンク（16）を介し、ホスト装置に送付して適切なデータ処理アルゴリズムを施し、プログラム制御装置のオペレーティングシステムの非強制的なデバッギングにモニタ装置を用いる。



【特許請求の範囲】

【請求項1】　プログラム制御を施したプロセッサ主体の装置の動作をモニタする方法において、
所定のプロセッサオペレーションシステムの事象を表わす当該装置の電子的信号を判定する過程、
前記プロセッサオペレーションシステム外の位置から、当該装置の信号バスにおける前記電子的信号の発生をモニタする過程、および、
当該信号発生の検出に際し、前記プロセッサオペレーションシステムの事象の所定の詳細を捕捉して報告する過程を備えたことを特徴とするプログラム制御装置モニタ方法。

【請求項2】　前記検出した事象に所定のメモリ位置に対するアクセスが含まれている場合に、当該メモリ位置に書込み、読出した事柄に関する情報の報告を前記捕捉して報告する過程に含んでいることを特徴とする請求項1記載のプログラム制御装置モニタ方法。

【請求項3】　事象発生点を指示するタイムスタンプの検出データへの加算を前記捕捉して報告する過程に含んでいることを特徴とする請求項1または2記載のプログラム制御装置モニタ方法。

【請求項4】　モニタの対象が前記プロセッサ主体の装置のデータバスであり、前記電子的信号が所定のデータワードであることを特徴とする請求項1記載のプログラム制御装置モニタ方法。

【請求項5】　モニタの対象が前記プロセッサ主体の装置のアドレスバスであり、前記電子的信号が所定のメモリアドレスであることを特徴とする請求項1記載のプログラム制御装置モニタ方法。

【請求項6】　事象の検出に際し、当該事象に引続く所定期間における全バス信号を捕捉することを特徴とする請求項4または5記載のプログラム制御装置モニタ方法。

【請求項7】　プログラム制御を施したプロセッサ主体の装置の動作をモニタする装置において、
当該プロセッサ主体の装置の信号バスに接続可能であって、特定のプロセッサオペレーションシステムの事象を表わす少なくとも当該装置の所定の電子的信号の発生を検出可能のフィルタ手段、
当該フィルタ手段に接続して事象の検出に際し、予定した詳細を書込み可能な事象キャッシュメモリ、
前記フィルタ手段に接続して、当該フィルタ手段を介し、前記事象キャッシュメモリから読出した事象の詳細を蓄積可能の事象データ蓄積メモリ、および、
前記フィルタ手段および前記事象データ蓄積メモリに接続してそれぞれの動作を制御するプロセッサ回路を備えたことを特徴とするプログラム制御装置モニタ装置。

【請求項8】　前記フィルタ手段が、所定の装置信号の配列から少なくとも一つの装置信号を制御可能に検出し得るプログラム制御可能の装置であることを特徴とする請求項7記載のプログラム制御装置モニタ装置。

【請求項9】　前記フィルタ手段に接続して、フィルタ構成を規定する情報を蓄積するフィルタ構成蓄積手段を備えたことを特徴とする請求項8記載のプログラム制御装置モニタ装置。

【請求項10】　前記プロセッサ回路が、捕捉した事象の詳細を、前記事象データ蓄積メモリにおける第1領域から読出し、処理して、他の領域に再度書込み得ることを特徴とする請求項7乃至9のいずれかに記載のプログラム制御装置モニタ装置。

【請求項11】　前記信号バスに接続可能であって、前記プロセッサ主体の装置の動作に割込みを行い得るデバッグ手段を備えたことを特徴とする請求項7乃至10のいずれかに記載のプログラム制御装置モニタ装置。

【請求項12】　前記プロセッサ回路、前記フィルタ手段および前記事象データ蓄積メモリに接続可能の双方向通信ポートを備えたことを特徴とする請求項7乃至11のいずれかに記載のプログラム制御装置モニタ装置。

【請求項13】　被モニタ装置オペレーションシステムの知識によって決定したデータ処理アルゴリズムを施すプログラム制御ホスト装置および双方向通信リンクを介して当該ホスト装置に接続した請求項11記載のモニタ装置を備えたことを特徴とするデバッギング装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、プログラム制御を施したプロセッサ主体の装置の動作をモニタする方法および装置に関するものである。かかるモニタは、プログラムのデバッギング、すなわちプログラム中の符号誤りの識別を容易にし、ソフトウエアのタイミングおよび特性の解析であるプロファイル能力を容易にする。

【0002】

【従来の技術】コンピータプログラムを書く場合に、ソフトウエア技術者は屡々デバッグツールを使用するが、このデバッグツールは、プログラムの個々の部分の正しい動作を立証し得るようにして、個々のプロセッサへの命令もしくはその各部の実施を可能にすることにより、プログラマがプログラムの進行全体を大幅に制御し得るようにする。

【0003】従来の強制的なデバッグツールは、プログラムの予定点に中断箇所を配置する相互作用的デバッグ効用を用いることであり、中断箇所に達すると、プログラムの進行を中断して系統の状態のテストが行われる。この中断によって問題が生ずるおそれがあり、プログラム制御の実施の中断は、時間的にクリティカルな動作の長い継続を中断するものであって、しからざる場合には、かかる動作中のプロセッサのアクティビティを解析することが必要となる。

【0004】従来のデバッグツールは、コンピュータシステム内の入出力ポートやメモリなどのサブルーチンの

操作とともに、プロセッサ内の記録値をテストして修正するが、コンピュータ内で進行するオペレーションシステムの存在を完全に無視している。かかるオペレーションシステムには、外部割込みの予期せぬ発生時に実行される符号とともに、符号のタイミングにクリティカルな部分が含まれており、この種の符号の実行はプログラム内の動作のみを追跡するとともに、キーボード入力やスクリーン表示などのオペレーションシステムの能力を利用する従来のデバッグツールによっては考えられないことである。

【０００５】デバッグツールは、コンピュータシステムが最初に設計されたとき以来、その概念を変えていない。新しい変形がCのような高レベルの言語を支持してはいるが、「プロセッサ」が実行する一連の動作を規定する「プログラム」の線に沿ったプログラムデバッギングのタスクをなお残している。高レベル言語デバッグシステムの例は、米国特許第 5,127,103号明細書に記載されているが、高レベル言語デバッグシステムは、モニタすべきプロセッサの知識に加えて、コンパイラの知識を必要とし、さらに、ジャンプや分岐サブルーチンに依らないシステム呼出しのような予期した言語の要求には順応しない事象が生じた場合に問題に遭遇することになる。

【０００６】

【発明が解決しようとする課題】従来のデバッグツールが直面する問題の例として、オペレーションシステムに対する呼出しをメモリのディスクセクタにロードする適用例では、従来のデバッグツールがそのシステム呼出しを単一のプログラム表現として取扱う。従来のデバッグツールは、システム呼出しを分割不能の事象として取扱い、メモリ領域のロードの成功、不成功に応じて動作システムがプログラムの実行をアプリケーションに戻したときに、次のプロセッサ命令を考慮するように動く。システム呼出しが近来のマルチタスクオペレーションシステムに対して行われた場合に実際に起こることは、プロセッサがディスクコントローラにあるメモリ領域にロードしたことを知らせ、次いで、そのロードが完結したことを知らせるまで同時に進行している他のアプリケーションに切換わることである。このディスクコントローラによるロードには、ディスクドライブの制御、プロセッサが他の命令を実行し得ないようにするシステムバスに対する要求、所要メモリアドレスのディスク領域に含まれるデータを蓄積するためのダイレクトメモリアクセス（ＤＭＡ）の使用、プロセッサの実行を再開させるシステムバスの開放、および、オペレーションシステムの他の領域に、ディスク領域のロードが完結したことを示すフラグを設定させて、元のオペレーションシステムの呼出しが復活するようにさせる割込みの発生が含まれる。

【０００７】従来のデバッグ技術の代替は、回路内エミュレータ（ＩＣＥ）であり、その一例が、１９９４年１月１８日発行の米国特許第 5,280,626号明細書に記載されている。この例においては、コンピュータシステム内のプロセッサを、プログラマの制御のもとにのみプロセッサを模倣してＩＣＥ装置に接続するテスト装置に置き換えてあり、ある面では従来のデバッガと同様であるが、プログラマがオペレーションシステムのコードに曝されるので、アプリケーションとオペレーションシステムとの命令間の区別がつかなくなる。

【０００８】他の周知技術は、コンピュータシステム内の点に接続して利用可能の信号のタイミングおよび状態の情報を発生させる論理解析器の使用であり、この場合、コンピュータシステムは複雑な信号発生器として取扱われ、論理解析器は、単に、コンピュータシステムの知識と先入観とに基づく解析のために、その信号をプログラマに報告するだけであり、これを補助するためには、ドン アトキンス著「６８０３０プログラム矯正における論理解析器インターフェース補助」ＥＤＮ誌１９８８年９月２９日刊，１８７～１９２頁に記載されているように、論理解析器インターフェースを使用することができる。通常のモニタ点における信号を利用可能にするのに加えて、上述のインターフェースは、通常モニタした信号から付加的な「クォリファイア」信号を発生させて、例えば「プロセッサ休止」、「例外保留」もしくは「命令実行」を指示することにより、プログラマを補助する。

【０００９】

【課題を解決するための手段】本発明の目的は、モニタするコンピュータシステムの正常動作には影響を与えない非強制的なモニタ方法を提供することにある。

【００１０】本発明の他の目的は、コンピュータオペレーションシステムを考慮してコンピュータシステムの実時間モニタを可能にすることにある。

【００１１】本発明によれば、所定のプロセッサオペレーションシステムの事象を表わす当該装置の電子的信号を判定する過程、前記プロセッサオペレーションシステム外の位置から、当該装置の信号バスにおける前記電子的信号の発生をモニタする過程、および、当該信号発生の検出に際し、前記プロセッサオペレーションシステムの事象の所定の詳細を捕捉して報告する過程を備えて、プログラムを施したプロセッサ主体の装置の動作を監視する方法が提供される。

【００１２】また、本発明によれば、当該プロセッサ主体の装置の信号バスに接続可能であって、特定のプロセッサオペレーションシステムの事象を表わす少なくとも当該装置の所定の電子的信号の発生を検出可能のフィルタ手段、当該フィルタ手段に接続して事象の検出に際し、予定した詳細を書込み可能な事象キャッシュメモリ、前記フィルタ手段に接続して、当該フィルタ手段を介し、前記事象キャッシュメモリから読出した事象の詳細を蓄積可能の事象データ蓄積メモリ、および、前記フ

ィルタ手段および前記事象データ蓄積メモリに接続してそれぞれの動作を制御するプロセッサ回路を備えて、プログラム制御を施したプロセッサ主体の装置の動作をモニタする装置が提供される。

【００１３】本発明の他の特徴は、特許請求の範囲に規定されているとおりであり、参照すべきである。本発明のモニタ方法によれば、デバッギングおよびプロファイリングが、ダイナミックバーチャートが割込みの瞬時周波数を示すように、あるいは、デバッグ用ハードウエアから受ける蓄積データを見れば判るように、実時間で行われる。また、特定の事象の選択により、その事象に対して渡したパラメータもしくは実行時間の表示が行われる。

【００１４】デバッグシステムにシンボルテーブルもしくはメモリマップを供給することにより、非強制的なデバッグシステムは、実行ヒストリ、もしくはシステムの失敗に引続く後戻りレコードを提供することができる。また、トリガ能力は解析の開始をトリガすべきアプリケーションの特定のサブルーチンを使用者が特定し得るようにする。ハードウエアが捕捉したデータは、後続の解析用もしくは比較用に蓄積しておくことができる。

【００１５】このデバッグシステムの、以上に説明した従来の方法に勝る主な利点は、つぎのように要約することができる。付加的なメモリの使用、非常ローディングもしくは実行時間割込みなど、モニタ装置からの妨害を受けることなく、コンピュータシステムに対してモニタリングを行うことができ、これは従来のデバッグツールではあり得ないことである。

【００１６】オペレーションシステムの知識を新たな装置から得たデータの説明に用い得るので、オペレーションシステムとアプリケーションとの相互作用を充分に探究することができ、これは従来の何れの方法でもあり得ないことである。完全なシステム、すなわちハードウエアの構成、オペレーションシステムおよび組合わせアプリケーションのデバッギングを実行することができ、これは従来のデバッグツールあるいはＩＣＥによってあり得ないことであり、デバッグデータの説明の自動化により、論理解析器によって見出したデバッギングの頼りにならない性質を除去することができる。

【００１７】

【実施例】本発明の方法及び装置の特別な用途は、これに限定されることはないが、オペレーティングシステムＯＳ－９の特性を利用するＣＤ－Ｉ装置のためのハードウェアデバッグツールを提供することである。本発明がどのようにして例えばオペレーティングシステムＯＳ－９に適用することができるかが当業者にとっては自明のものであっても、以下の記述はこのようなデバッグツールについてのものである。

【００１８】これまでのデバッグ方法の効果を減らすタイミングの問題は、ＣＤ－Ｉプレヤーがマルチタスクの環境にあるために起きる。アプリケーションが実行されている際には、プロセッサはディスプレイを更新するとか或いはＣＤドライブから読み出すといった別のタスクを実行することもできる。更に、システムに接続された装置（例えばローラーコントローラ）は活性化されるとプロセッサに対して割込みを発生し、これらはプロセッサが適切なアクションをとるためのアクティビティを要求する。このマルチタスク環境の影響は、従来のデバックを用いたアプリケーションの解析が全てのプロセッサのアクティビティを示さないことである。ＣＤ－Ｉタイトル（例えばアクションゲーム或いはディジタルビデオタイトル）の複雑性が増えるにつれて、アプリケーションはより大きなプロセッサ時間を要求するようになり、アプリケーションそのものではなく、完全なプロセッサアクティビティを解析することが更に重要になって来る。

【００１９】非強制的なデバッギングは、ＣＤ－Ｉプレーヤが実際にコードを実行している間、このＣＤ－Ｉプレーヤが何を実行するかを推論する。この推論はＣＤ－Ｉプレーヤ中の拡張バスに得られる信号をモニタすることにより達成される。次に、アルゴリズムを用いてバスのアクティビティを出力し、特定の事象を検出しうるようにする。これらの事象は、メモリからの命令の取出しや、システムスタックへの書込みのような他のバスアクティビティをモニタすることにより検出される。このバスアクティビティには以下のことが含まれている。

【００２０】システムの呼出し。システムに渡されたパラメータを検出するようにするシステム呼出し処理中システムスタックにおけるすべてのレジスタの記憶。タイマ又はＣＤプレーヤのような装置からの割込み。割込み中に成されたシステム呼出し又は他のシステム呼出し。割込みからの復帰。サブルーチン呼出しおよび復帰。定義済メモリアクセス。

【００２１】従来の強制的なデバッキング中、ブレークポイントが一般にサブルーチン呼出しやシステム呼出しの前又は後に入れられる。これにより、正しい呼出しが成されたかや、所望のパラメータが渡されたかをプログラマが確認しうるようにする。これにより、非強制的なデバッギングがサブルーチン呼出しやシステムの呼出しを検出しうるようにすることや、プレーヤを停止することなくシステム呼出しに対しパラメータを渡しうるようにすることを特に有効にする。更に、システム呼出しが（オペレーティングシステム内にあるものの為に強制的なデバッギングではプログラマに得ることができない）他のシステム呼出しを実行しうる為、非強制的な方式によればデバッギングの一層の細分化が得られる。

【００２２】非強制的なデバッギング中は、プログラマはデバッギングホストに関する対話式のデバッギングツールを用いてプロセッサアクティビティを解析しうる。特定のアクティビティ（例えば割込み又はシステム呼出

しの発生）の瞬時的レベルを実時間で決定でき、記憶されたデータの解析により、システム呼出しに対し渡されたパラメータを検査したり、サブルーチンの逆戻りをCD－Iプレーヤシステムの故障に続いて行ったりさせる。

【0023】非強制的な割込み、サブルーチン呼出しおよびシステム呼出し検出とタイミング情報とを組合せることにより簡単なプロファイリングを実効しうる。これにより、プロセッサが所定のタスクを実行するのに費やした時間の部分を決定でき、所望に応じ調整しうる。この情報によりプログラマは速度に対するコード区分を最適化しうる。

【0024】アプリケーションはOS－9においてバスアクティビティの標準パターンを生じるTRAP＃0命令を用いてシステム呼出しを行う。これにはTRAP＃0演算コードおよびベクトル番号（O_x 4E40）のプログラムメモリからの読出し、ショートスタックフレームのシステムスタックへの書込み、ベクトルの内容の読出しおよび当該アドレスへのジャンプが含まれる。これに続いて、すべてのレジスタがシステムスタックにセーブされ、システムファンクションコードがTRAP命令に続くメモリ位置から読出される。システム読出し（これには他のシステム呼出しを含めることができる）が処理された後、レジスタが回復され、トラップが復帰する。システム中の各レジスタの内容は、デバッギングシステムのモニタ用ハードウェアがTRAP＃0のオペレーションの開始を識別しうる限りシステム呼出しが生じる度に決定しうる。

【0025】パラメータは、システム呼出しが成される前にこれらパラメータを所要のレジスタ内にローディングすることによりシステムファンクションに渡される。システムスタックに対するすべてのレジスタの書込みを解析することによりこれらのパラメータを識別しうる。システム呼出しファンクションコードはこれをプログラムメモリから読出した際に決定でき、トラップの開始および終了時における時刻表示により、プロセッサがそのサービスにどの位の時間を費やしたかを表示することができる。TRAP＃0命令と同期する機能を有するシステムは極めてパワフルなデバッギングツールを提供し且つ第1レベルのプロファイリング機能を提供すること明らかである。デジタルビデオタイトルの製作やこれに関連するCD－Iプレーヤに対する計算上のロードが高まるにつれ、CPUアクティビティを非強制的にプロファイリングしうるようにする重要性が高まってきている。

【0026】多くのCD－Iプレーヤの心臓部にある68070マイクロプロセッサは先取り命令を用いて効率を高めている。このことは、プロセッサが現在の命令を復号している際に命令符号がメモリから読出されることを意味する。このようにすることにより、マイクロプロセッサのバスアクティビティが、プログラムを構成する命令符号の列に直接相関関係をもたなくなるという効果が得られる。システム呼出しにこのことを用いると、ショートスタックフレームが必ずしもTRAP＃0演算コードの取出しに追従しなくなる。その理由は、この命令はプロセッサが前の命令を複合している間に取出される為である。メモリ位置のクリアのような前の命令の効果は、トラップ命令が取出される後までシステムバスに現われることができない。

【0027】TRAP＃0に続く事象の列の中に割込みが生じる場合には、デバッギングハードウェアはプレーヤとの同期を失うおそれがある。しかし、割込みサービスルーチンはしばしばシステム呼出し自体を行う為、割込みが生じた時を信頼的に検出しうるようにすることが重要である。

【0028】バスアクティビティのみをモニタすることによっては割込みの発生を決定するのは困難である。外部割込み要求や応答ラインはモニタすることができるが、同じく割込みを生じるUARTおよびDMACのようなサブシステムには68070マイクロプロセッサが組込まれている。これらの割込みが処理中であるということを表わす外部信号はない。

【0029】割込みがペンディング中であることをプロセッサが検出すると、プロセッサは現在実効されているタスクの優先度と割込みの優先度とを比較し、割込みの優先度が高い場合のみこの割込みのサービスをする。プロセッサが割込みのサービスを開始すると、ベクトルがこれを生じた装置から決定され、ショートスタックフレームがシステムスタックに書込まれ、割込みベクトルの内容が読出される。次に、ベクトルに記憶されたアドレスから実効が継続される。内部的に発生される割込みは自動ベクトル化されており、従ってベクトルの取出しは実行されない。

【0030】先取り命令および割込み命令は、これらがない場合にデバッギングシステムに有効情報を与えうるバスアクティビティの予測可能な列を害するおそれがある。従って、事象の検出は個々のバスサイクルを頼っている。

【0031】TRAP＃0に続くバスアクティビティを本明細書の終りに記載したリスト1につき説明する。このリスト1においては、それぞれの列が左から右に、状態番号、プロセッサの読出し(Read)又は書込み(Write)、アドレスバス内容、データバス内容および注釈又はコード分解を示す。バス状態はアドレスストローブ信号（ASN）の立上がり縁でラッチされることに注意すべきである。

【0032】TRAP＃0は、プログラムがシステム呼出しを行うことも望む際に実行される。代表的なコード断片は以下の通りである。

【数1】

アドレス 0x1AF5A 演算コード 0x426A0008 アセンブラ clr.W8(a2)
0x1AF5A 0x4E400084 cs9 I$Open

この場合、クリアワード命令にシステム呼出しが続く。アドレスＯｘ１Ａ１Ｆ５ＥはＴＲＡＰ＃０演算コード（Ｏｘ４Ｅ４０）を含み、次のアドレスはシステムファンクションコード

【数２】

（I＄Open に対し Ｏｘ００８４）

を含む。実行に続くバスアクティビティをリスト１に示してあり、このリストを参照されたし。

【００３３】状態１～４：リスト１から明らかなように、先取り命令によりバスアクティビティをアセンブラコードに対し異なる順序にする。トラップ演算コードは、プロセッサがレジスタａ２の内容に８を加えることにより有効アドレスを計算している間に取出される。命令が取出された後にのみ、メモリの書込みが行われる。

【００３４】状態５～８：トラップ＃０演算コード（状態１～４）の読出しに続いて、ショートスタックフレームがシステムスタックに記憶される。ショートスタックフレームは、ＴＲＡＰ＃０に対するベクトルのオフセットＯｘ８０を含むヘッダと、ＴＲＡＰ＃０命令が位置するプログラムカウンタ（ＰＣ）と、状態ワードとより成る。ヘッダワードにおける最上位からのニブルが零であることはショートスタックフレームを表わす。

【００３５】状態９～１７：所定のベクトルが読出され、このベクトルに記憶されているアドレスに飛越しが行われる。プロセッサがベクトル内容から演算を開始した後、２ワードがスタックにプッシュされ、次に更なる飛越しが行われる。命令の先取りの為に、更なる飛越しの演算コードと、オペランドの取出しとの間で２ワードのプッシュが生じること明らかである。

【００３６】状態１８～１９：すべてのデータおよびアドレスレジスタ（スタックポインタａ７を除く）がスタックにセーブされる。

【００３７】状態２０～３３：スタックにおけるアドレスレジスタの記憶。

状態３４～４９：スタックにおけるデータレジスタの記憶。

【００３８】状態５０～６４：オペレーティングシステムのベース（基底）アドレスの読出し。ａ５からの最終読出しによりシステム読出しファンクションコードのアドレスを、記憶されたＰＣにより指摘されているその記憶位置からショートスタックフレーム内にロードさせる。演算コードおよびこの命令に対するオペランドの取出しには前の命令からの書込みが割込まれていることに注意すべきである。

【００３９】状態６５～６６：ファンクションコードの読出し：これはＴＲＡＰ命令後に規定されたワードである。ＴＲＡＰ＃０を除くベクトルが読出された点（リスト１の状態９）から６４サイクルに関するデータをカートリッジがセーブすると、このセーブされたデータによりカートリッジプロセッサが以下の情報を決定するようにする。

－状態２０～４９からのすべてのレジスタの内容。

－状態６６におけるデータバスの値からのシステム呼出しのファンクションコード。

－システム呼出しが行われたアドレス（このアドレスからシステムコードが状態６６で取出された）。

【００４０】次に割込みの処理を考慮するに、ＣＤ－Ｉプレーヤにおける６８０７０プロセッサが以下の多数のソースからの割込みを受ける。

－ＩＮ２，ＩＮ４，ＩＮ５およびＮＭＩ復号割込み信号。

－ＩＮＴ１およびＩＮＴ２ラッチ割込み信号。

－組込みタイマ。

－ＲＳ２３２受信および送信。

－ＤＭＡチャネル１および２。

【００４１】ＩＮ２，ＩＮ４およびＩＮ５復号割込み信号は拡張バスにおけるこれらの応答ラインで得られるが、これらの信号は割込みがいつ生じているかの完全な表示は与えない。割込みはプロセッサ内のタイミングおよびＤＭＡ回路によっても発生され、プロセッサがこれらソースの１つからの割込みをサービス（処理）していることの外部表示はない。

【００４２】割込み（これが内部であろうと外部であろうと）がプロセッサに送られると、プロセッサはこの割込みが現在のタスクの優先度よりも高い優先度を有している場合のみこの割込みのサービスをする。この割込みの優先度が低い場合には、ベクトルが装置から読出され、ショートスタックフレームがシステムスタックに記憶され、ベクトルアドレスの内容が読出され、このアドレスに飛越しが行われる。この例は、割込みが状態２でサービスされているリスト２に示されている。スタックフレームヘッダにおけるベクトルオフセットはＯｘ２００であり、これはロングワード内容を有する各ベクトルエントリの為に４倍されたベクトル（Ｏｘ８０）である。

【００４３】ＣＭＡＣからのような外部割込みが生じた場合には、割込みベクトルの取出しはない。その代わり、内部オートベクトル割込みが発生される。リスト３のコード抽出はシステム呼出しの状態５１（システムスタックへの全レジスタ記憶の直後）で生じるような割込みを示している。スタックフレームに記憶されたプログラムカウンタ（ＰＣ）の値は、状態５０における命令が取出されているが実行されていないアドレスであること明らかである。

【００４４】リスト２および3のコードセグメントは、外部割込みが割込みベクトルを割込み装置から取出し、一方、内部割込みがオートベクトル化されることを示している。前述したように、６８０７０プロセッサは先取り命令を実行し、ペンディング中の割込みのサービスへの切換えが行われるのはこの先取り命令の後である。この切換えが行われると、演算コード命令が捨てられ、この命令が取出されたアドレスがスタックフレーム内に記憶されている更なるアドレスの値となる。従って、最低で１Ｋバイトのメモリにおける記憶位置からの割込みベクトルの取出しにより割込みが検出される。

【００４５】トラップすなわち割込みの終了時にはＲＴＥ（例外からの復帰）命令が実行される。この命令の瞬時に続くバスアクティビティはリスト４に示す通りにすることができる。ＲＴＥ命令の実行に続いて、ショートスタックフレームがシステムスタックから除去され、スタックフレーム内に記憶されているアドレスから次の命令が取出される。

【００４６】サブルーチンが呼出されると、プログラムカウンタ（ＰＣ）の値がスタックに記憶される。サブルーチンからの復帰が行われると、スタックにおけるトップ値が除去され、プログラムカウンタの新たな値として用いられる。サブルーチンからの復帰は、値Ｏｘ４Ｅ７５を有する“サブルーチンからの復帰”演算コード（ＲＴＳ）を読出すためのフィルタリングにより容易に検出しうる。サブルーチン呼出しは検出するのがわずかに困難である。その理由は、以下の幾つかのバージョンがある為である。

【００４７】－ＪＳＲ（ａ０）のような内部発生アドレスを有するジャンプサブルーチン（ＪＳＲ）が、サブルーチンが記憶されている位置（ａ０，ａ１，等）に応じて異なる演算コードを有する。命令はＯｘ４Ｅ９０～Ｏｘ４Ｅ９７の範囲の単一ワードより成る。この種類のサブルーチン呼出しに続くバスアクティビティをリスト５に示す。

【００４８】状態０においては、サブルーチン呼出し演算コードが読取られ、状態１では、呼出されたサブルーチンの第１演算コードの有用な読取りが行われる。その場合には、命令が復号されるとともに、状態２および３で現在のＰＣがスタックに書込まれる。状態４においては、新たなサブルーチンで命令の処理が継続する。事象の検出は、範囲Ｏｘ４Ｅ９０乃至Ｏｘ４Ｅ９７におけるワードの読取りに対し、フィルタリングによって行われ、単一ＣＤ－Ｉバスサイクルだけ遅延するので、状態１の期間中に起こることになり、これにより、呼出されたサブルーチンのアドレスが、事象データの一部として蓄積されて、蓄積されているシンボルテーブルと比較されることが可能となる。

【００４９】ＪＳＲ６（ａ０）のような付加的オフセットを用いて計算したアドレスを付したジャンプサブルーチン（ＪＳＲ）：これは、サブルーチンが蓄積されている場所（ａ０，ａ１など）によって異なる演算コードを有するとともに、オフセットを特定すべき演算コードに引続くオペランドワードも有している。命令は、範囲Ｏｘ４ＥＡ０乃至Ｏｘ４ＥＡ７における単一ワードからなっている。事象検出は、この領域におけるワードの読取りに対するフィルタリングによって行われ２ＣＤ－Ｉバスサイクルだけ遅延するので、オフセット読取りを超えてスキップし、新たなアドレスのサブルーチンからの演算コード取出し期間中に再度起こる。これは、いわゆるサブルーチンのアドレスが事象データの一部として蓄積されて、蓄積されているシンボルテーブルと比較されるのを可能にする。

【００５０】８ビット変位を伴ったブランチサブルーチン（ＢＳＲ）は、その変位が演算コードワードの低位のバイトに蓄積されるので、変位量に応じて異なる演算コードを有している。したがって、命令は、範囲Ｏｘ６１０１乃至Ｏｘ６１ＦＦにおける単一ワードからなっており、この型のサブルーチン呼出しに引続くバスアクティビティを次のリスト６に示す。

【００５１】状態０においては、サブルーチンブランチ演算コードを読取り、状態１では、呼出されたサブルーチンの第１演算コードの有用な読取りが行われる。その場合には、命令が復号されるとともに、状態２および３で現在のＰＣがスタックに書込まれる。状態４においては、新たなサブルーチンで命令の処理が継続する。事象の検出は、範囲Ｏｘ６１０１乃至Ｏｘ６１ＦＦにおけるワードの読取りに対し、フィルタリングによって行われ、単一ＣＤ－Ｉバスサイクルだけ遅延するので、状態１の期間中に起こることになり、これにより、呼出されたサブルーチンのアドレスが、事象データの一部として蓄積されて、蓄積されているシンボルテーブルと比較されることが可能となる。

【００５２】１６ビット変位を伴ったブランチサブルーチン（ＢＳＲ）は、ブランチ演算コードに引続く１６ビットオフセットワードとともにＯｘ００と規定した上述の場合には、８ビットの変位を有している。したがって、命令は、１６ビットオフセットが引続くＯｘ６１００の単一ワードからなっている。事象検出は、Ｏｘ６１００ワードの読取りに対するフィルタリングによって行われ、２ＣＤ－Ｉバスサイクルだけ遅延しているので、オフセット読取りを超えてスキップし、新たなアドレスのサブルーチンからの演算コード取出し期間中に再び起こる。これにより、読出されたサブルーチンのアドレスが事象データの一部として蓄積されて、蓄積されているシンボルテーブルと比較されるのを可能にする。かかるバスサイクルの期間中、アクセスがＤＭＡに対して行われるが正常な処理ではない。

【００５３】本発明を実施するモニタ装置は、ディジタルビデオカートリッジと同様に、ＣＤ－Ｉプレーヤの開

発もしくは製造に適合したカートリッジの形態をなしている。カートリッジはプレーヤバスアクティビティの収集、解析および蓄積並びに結果を得るためのホストデバッグシステムとの連絡に従事し、収集および解析のアルゴリズムの実時間構成を許すものであり、ホストは、適切にプログラム制御したＰＣなどのコンピュータとする。

【００５４】カートリッジのブロック線図を図１に示す。デバッグカートリッジは、６８０２０プロセスコアをベースとしたモトローラ社ＭＣ６８３４０型マイクロプロセッサ１０を模して設計されており、付加的集積サブシステムを伴った高性能３２ビットプロセッサであり、２チャネルＤＭＡコントローラ、２個のタイマ／カウンタ、２チャネルＵＳＡＲＴ（ユニバーサルシンクロナス／アシンクロナスレシーバ／トランスミッタ）、並びに、システム保護を行い、チップセレクトおよび待ち状態を評価し、クロックシンセサイザ、外部バスインターフェースおよび裁定マネージャを含むＳＩＭ（システムインテグレーションモジュール）を包含している。この装置は２５ＭＨｚまでの周波数で動作する。

【００５５】ＳＩＭは、四つの外部チップ選択信号が、外部メモリ（ＲＯＭ蓄積、ＲＡＭ蓄積、プログラム制御可能の論理構成およびデバッガの場合のプログラム制御可能の論理内部レジスタ）とそれぞれ簡単にインターフェースし得るようにする。三つまでの待ち状態をＥＰＲＯＭなどの比較的低速のデバイスに対して有用なようにプログラム制御し得るとともに、ダイナミックバスサイズも支持する。別様にプログラム制御されない限り、各バスサイクル毎に行われる「グローバル」チップセレクトは、システム初期化が起こる前のブートＲＯＭに対する補助的アドレス復号化の必要をなくしている。

【００５６】ファームウエアは単一の１２８キロバイト２７Ｃ１０２４ＥＰＲＯＭデバイス１２に蓄積され、グローバルチップセレクトは、三つの待機状態を有する１６ビットポートとして動作するようにプリセットされており、２５メガヘルツのクロックレートで１８０ナノ秒のアクセス時間に対応しているが、ファームウエアは、次の各要素からなっている。

－ブートデータおよび初期化コード：

－カートリッジのオペレーションを証明するためのパワーオンセルフテスト：

－スピード用ＲＡＭにコピーした直列ポート基本要素その他のＢＩＯＳルーチン：

－基本的なコマンドラインインタープリタ（ＣＳＩ）および対応するカートリッジ開発用ルーチン：

－ロードして実行すべきローダコードおよびオペレーションシステム：

－使用中のロード時間を最小にするためのライブラリフィルタ構成ファイル：

【００５７】適切にも、直列リンク上のロードを減少させるとともに、フィルタのプログラム制御のレートを増大させるためにＥＰＲＯＭに蓄積した以下に説明するようなフィルタ構成ファイルをカートリッジが多数所有している。

【００５８】デバックタスクはかなりの量の高速揮発性メモリを必要とするが、これを最も容易に提供する方法は、周期的な回復を必要とせず、極めて高速のアクセスレートで利用し得る故のＳＲＡＭ１４の使用であり、４メガバイトまでのＳＲＡＭデバイスを１メガバイトモジュール中のカートリッジに付加することができ、カートリッジの初期化中に、利用可能のメモリを決定して、つぎの各領域間でこのメモリを区分けする。

－割込みベクトル、スタック等のシステムデータ領域。

－直列ポート基本要素、バッファ等のＢＩＯＳルーチン領域。

－コード、データ、バッファ等のオペレーションシステム領域。

－事象キャッシュから読出した未処理の事象データ蓄積領域。

－バックトレースバッファ、事象スタック等の処理済み事象蓄積領域。

フィルタ設計が直列リンクを超えて重ねてロードされるのを防ぐロード済みフィルタ構成ファイル。

【００５９】アドレス復号化は、二つの高速プログラム制御可能論理デバイスによって行われ、全アドレス領域に亘ってバイトもしくはワードのアクセスを可能にするが、高性能のＳＲＡＭデバイスは高価であるので、安価な方のＤＲＡＭデバイスを、性能低下のおそれはあるが、使用することができる。

【００６０】高速度直列リンク１６は、カートリッジとデバッグホストとの間の連絡用に、６８３４０マイクロプロセッサの両直列通信チャネルを用いて設定され、その第１チャネルは、標準ターミナルに対して十分な二重通信を設定するが、これは、カートリッジＣＬＩおよび対応するサブルーチンと共同して、カートリッジのオペレーションおよび特性を解析するとともに、カートリッジのサブシステムをテストするのに使用する。連絡は、ＲＴＳおよびＣＴＳの信号を用いて達成されるフロー制御により、９６００ボーで行われる。

【００６１】第２チャネルは、適切なプロトコルを用いて、デバックホストとの充分な二重通信を設定するが、これは、デバッグホストと構成および事象の報告に関与するカートリッジとの間の通信全部に用いられる。この通信は、ＲＴＳおよびＣＴＳの信号を用いて達成されるフロー制御で１９２００ボーで行われる。

【００６２】デバッグホストで進行しているソフトウエアの制御のもとに、事象もしくは事象の組合わせが特定のデバッグファクションを達成するためのメニューから選ばれ、ついで、ホストソフトウエアは、ほぼ８キロビットの大きさの適切な構成ファイルを、そのときプログ

ラム制御可能の事象フィルタ１８を構成するカートリッジに、そのファイルがそのカートリッジ中に既に存在していなければダウンロードする。

【００６３】プログラム制御可能のフィルタは、９０００ゲートフィールドプログラマブルゲートアレイ（ＦφＧＡ）を用いて完成されるが、これは、所定のファンクションを達成するために構成して接続し得る３２０個の組合わせ論理ブロック（ＣＬＢｓ）を含有している。多数のフィルタ設計が、そのデバイスが与え得る範囲の機能性に応じて必要とされるが、その設計のすべてに、同一モジュール、すなわち、事象フィルタ、キャッシュＲＡＭメモリコントローラおよびタイムスタンプ発生用手段が存在している。

【００６４】多数の事象フィルタがＣＤ－Ｉ拡張バス上の信号をモニタして、条件が整ったときにトリガするが、これは、ＤＭＡサイクルが進行しつつあるとき、特定のメモリアドレスをアクセスしたとき、システム呼出しが発生したとき、サブルーチンが呼出され、あるいは復帰したとき、割込みが発生し、あるいは復帰したとき、などに起こる。フィルタ内のレジスタはフレキシビリティの手段を提供するためのものである。また、特定のフィルタが不能になることがあり、メモリアドレスアクセス用のフィルタには、使用者が関心を有するアドレスを保持したレジスタが含まれていることがある。

【００６５】事象が検出されると、一，二の事象が起こり得る。事象検出回路は、その事象に組合わさったデータバス、アドレスバス、タイムスタンプ等のパラメータからなる事象フレームを事象キャッシュメモリ２０にセーブするか、後段の詳細な解析用に、後続するバスアクティビィ全部の事象キャッシュへのダンピングを開始する。事象キャッシュはＦＩＦＯ蓄積領域としてオペレートする。ＦＧＰＡ内のキャッシュ制御回路は、事象発生の都度の事象データの書込み用、もしくはプロセッサによる事象データの読出し用のアドレス発生を維持する。

【００６６】ある事象が発生した場合に、事象キャッシュメモリ２０に書込まれる事象フレームは、タイムスタンプを含んでおり、これは、マイクロ秒オーダのレゾリューションを与える増分カウント、もしくは、ＣＤ－Ｉプレーヤ映像回路のフレーム／ライン情報をベースにしたタイムスタンプであり得る。

【００６７】プログラム制御可能のフィルタ１８は、１２８キロバイトの大きさとするのが好適な事象キャッシュメモリ２０に対するあらゆるアクセスを制御する。事象キャッシュメモリに関連したアドレスデータおよび制御信号は、プログラム制御可能のフィルタ１８によって管理されている。

【００６８】このメモリへの書込みは、事象が発生した後に発生し、事象フレーム、もしくは事象に引続くＣＤ－Ｉプレーヤ拡張バスアクティビティのダンプからなっている。また、このメモリからの読出しは、プログラム制御可能フィルタ論理内のレジスタの読出しによって行われ、これは、プロセッサ１０とキャッシュメモリ２０との間の直接接続は存在せず、したがって、書込みレイテンシーは存在しないことを確実にしており、書込みタイミングがＣＤ－Ｉ拡張バスの高速度の故にクリティカルであるから、重要である。諸信号は、ＦＩＦＯバッファの「充満」を指示するプロセッサにとって利用可能のものである。

【００６９】プログラム制御可能のコミュニケーション論理デバイス２２（４２００ゲートＦＰＧＡ）は、マップ可能のＥＰＲＯＭ用のマップレジスタおよびアドレスデコード、並列コミュニケーション回路、並びに割込み発生回路を含めて、デバッグ環境にとって重要な多数のファシリティを提供している。

【００７０】このモニタカートリッジは、ＣＤ－Ｉメモリスペース中にマップした２個の１６キロバイトＥＰＲＯＭを適切に備えており、このＥＰＲＯＭのスペースアドレスは、その値をコミュニケーション論理内のレジスタに書込むことにより、このカートリッジが設定する。典型的には、このＥＰＲＯＭは多数のＯＳ－９モジュールを含んでおり、ベースアドレスをレジスタに書込んだ後に、ＣＤ－Ｉプレーヤをリセットして、このモジュールの位置決めをプレーヤの初期化中に行うことができる。

【００７１】コミュニケーション論理は、ＣＤ－Ｉプレーヤに割込みを発生させて、ベクトルを供給することができ、このベクトルは、コミュニケーション論理内のレジスタから得られる。これは、ホストソフトウエアのコマンドのもとに、あるいは、特殊な事象が検出された場合に、ＣＤ－Ｉの正常なオペレーションが割込まれるのを可能にする。このベクトルは、望むならば、マップ可能のＥＰＲＯＭメモリからのＣＤ－Ｉプレーヤのジャンプを起こさせることができ、このモニタカートリッジは、このようにして、従来のデバッギングを非強制的なバッギングと同様に実行することを可能にする。

【００７２】ＣＤ－Ｉとデバッグホストもしくはカートリッジとの間の高速度データ転送のために、コミュニケーション論理は、双方向並列コミュニケーションリンクを提供するが、これは８個の双方向データ信号とＳＴＲＯＢＥおよびＡＣＫフロー制御信号とのために必要なインターフェースを提供することになる。

【００７３】３２キロバイトＥＰＲＯＭ蓄積領域２４は、ＣＤ－Ｉプレーヤメモリマップに含まれるべきＯＳ－９モジュール用に設けてあり、これは従来の「ブレイクおよび解析」の特徴を、適切なデバッグ用ＯＳ－９モジュールに書込みを行うことにより補充し得るようにしている。ＣＤ－Ｉプレーヤと、プログラム制御可能のコミュニケーション論理によって設けるがパイプのようなＣＤ－Ｉオペレーションシステムに現れる初期のコミュニケーション能力を用いたカートリッジとの間に、他の

モジュールによって高レベルコミュニケーションインターフェースを設けることができ、さらに他のモジュールにより、初期の並列コミュニケーション能力を仮想ディスクのようなＣＤ－Ｉオペレーションシステムに現われさせることもできる。マップ可能の蓄積用のアドレス復号化は、コミュニケーション論理により、その論理内のレジスタによってベースアドレスを設定するのと同時に遂行する。

【００７４】カートリッジ用並列入出力ポート２６は、デバッギングホストとＣＤ－Ｉプレーヤとの間に、そのホストが指示する高速双方向コミュニケーションの能力を提供するが、すべてのホストが書込みと同時に並列ポートを介して読出しを支持するわけではないので、方向の制御は制御信号を用いたホストによって管理される。

【００７５】システム呼出しおよび割込みに関係のあるデータの検出、解析および蓄積に適応し得るハードウエアは、バスアクティビティの解析行うために、６８０７０信号に同期する必要があるが、これは信号ＡＳＮの立上がりエッジにバスの状態をラッチすることによって達成される。この信号は、そのバスからデータを受取り、もしくは、そのバスにデータを送り込むバスサイクルの終端でプロセッサによって表わされる。読取りおよび書込みのタイミング図を図２に示す。

【００７６】オペレーションの期間中、カートリッジは、アドレスデータおよびコントロールの各バス上の信＊号を受取り、必要な命令もしくは事象が検出されたときに６８３４０に情報を与え、これにより、プレーヤの拡張バスとデバッガ中のプロセッサとの間に大幅のフィルタリングを導入し、プロセッサが自由に事象処理およびホストコミュニケーションを管理し得るようにする。

【００７７】図３に示すように、プログラム制御可能の事象フィルタ１８（図１）は、プレーヤのアドレスデータおよびコントロールの各バス用にそれぞれ比較器３２および遅延要素３４を従えたバッファ回路３０からなり、バスアクティビティの特定のパターンを検出し得るようになっており、制御回路（図示せず）もキャッシュメモリ２０（図１）の管理用に設けてある。解析回路の説明のために、バッファ回路の各出力には、ＬＡ（ラッチドアドレス）、ＬＤ（ラッチドデータ）およびＬＣ（ラッチドコントロール）の各ラベルを付してあり、ＬＣの主要要素はＬＲＷＮ（ラッチドリード／ライト）である。この各ラベルは、ｎサイクル遅延した比較器の出力に付されており、ｎはＡＳＮ信号が支配する遅延設定手段３６によって設定された整数値である。したがって、「ＬＡ３＝０ｘ１２３４５６」はアドレスバスが１２３４５６の１／６０の値をとったときから３バスサイクル後に現われるブーリアン信号を表わす。トリガのオペレーションの例としては、つぎの各事象の検出が考えられる。

－スタックフレーム（トラップすなわち割込み）の発生 ──事象コード＃１
－スタックフレームの除去（例外の終了） ──事象コード＃２
－スタック上の全レジスタをセーブする命令 ──事象コード＃３
－システム呼出しファンクションコードを取出す命令 ──事象コード＃４

【００７８】プロセッサは、目下活動中の例外のスタック、例外を解析するためのバッファおよびその発見を報告して指令を受入れるためのホスト管理スキームを有し、正常なオペレーションの期間中、ホストと連絡して、バッファ中で発見した事象を処理する。これらの事象はヘッダとデータ領域とからなっており、ヘッダは事象の種類を規定し、事象の種類によって決まる大きさのデータ領域は、事象に関連した付加的データを含んでいる。例えば、スタックフレームの発生に関連したデータ領域には、例外が生じた命令アドレス、例外が生じた時点のシステムスタックポインタおよびスタックフレームに蓄積された状態レジスタが含まれる。これに加えて状態カウンタおよびタイミング情報の少なくとも一方を蓄積することができる。事象バッファ内の情報の処理に際し、プロセッサは、どの事象を例外スタック中に位置させるのが必要か、所定の例外は終了したか、ホストに報告する必要があるのは何か、プロセッサはデータをどのように処理しているか（例えばバッファがオーバフローしかけているか）などを解読することができる。

【００７９】スタックフレーム発生（事象コード＃１）の検出は、トラップすなわち割込みが発生したことを示すが、その条件はリスト７に示すとおりである。右側の条件は、少ない個数のゲートによって容易になし得る比較およびマスキングのみを必要とすることに注目されたい。このシーケンスの検出に際し、フィルタはプロセッサに対する割込みを発生させ、プロセッサはつぎの各情報を読取ることができる。

－ＬＤ６が例外のベクトルを提供する。

－ＬＡ６が、例えば、例外の前にシステムスタックポインタを提供し得る。

－ＬＤ５およびＬＤ４が、ともに例外が起こった例外アドレスを提供する。

－ＬＤ３が状態レジスタを提供する。

－ＬＤ２およびＬＤ１が、ともに例外ベクトルの値を提供する。

プロセッサは、一旦この情報をフィルタから読取ると、この情報を含んでいる事象バッファに新たなエントリを発生させる。

【００８０】スタックフレーム除去（事象コード＃２）の検出は、プロセッサが例外の処理を終了したことを示し、プロセッサは、次の各情報を読取ることができる。

－ＬＤ４が復帰しつつある例外のベクトルを提供する。

－ＬＡ４がスタックフレームの始端のアドレスを提供する。
－ＬＤ２およびＬＤ１が、ともに例外が起こった例外アドレスを提供する。
－ＬＤ３が蓄積されている状態レジスタを提供する。
プロセッサは、この情報をもった新たな事象バッファを再度創作することができる。
【００８１】マシンレジスタをセーブする命令（事象コード＃３）の検出は、レジスタがシステムスタック上でセーブされる都度、その内容を逐一捕捉させる。このことは、トラップコードの期間中に生じ、システム呼出しに応じたパラメータに対するアクセスを提供する。システム呼出しはプログラムの実行中の有意の時点で生ずるので、この事象の検出は、ＣＤ－Ｉアプリケーションのデバッグに対して有効なツールを提供することになる。この事象に対する条件はリスト９に示すとおりである。
【００８２】このシーケンスの検出に際し、フィルタはプロセッサに対して割込みを発生させ、プロセッサはつぎの情報を読取ることができる。
－ＬＡ１が命令の始端のアドレスを提供する。
【００８３】プロセッサは、レジスタが逐一システムスタックに書込まれるのをウォッチすることができ、さらに、新たな事象バッファエントリに蓄積することができる。
【００８４】システム呼出しファンクションコードを取出す命令（事象コード＃４）の検出は、どのシステムファンクションが呼出されているかをプロセッサに識別させる。このことはトラップコードの期間中に起こり、このことに対する条件はリスト１０に示すとおりである。
【００８５】このシーケンスの検出に際し、フィルタはプロセッサに対して割込みを発生させ、プロセッサはつぎの情報を読取ることができる。
－ＬＡ１が命令の始端のアドレスを提供する。
－ＬＡ０がファンクションコードのアドレスを提供する。
プロセッサは、この情報を新たな事象バッファエントリに蓄積する。
【００８６】プロセッサは、ＦＩＦＯに基づく事象バッファ２０に蓄積された事象群について考察する。その事象バッファ２０は、例外タイミングの不均一な性質が解析に先立って消滅されるようにする。一例として、バッファ中の次の事象がコード＃２の事象である場合に、プロセッサが例外の終りに行う解析について考察するに、プロセッサは、除去したスタックフレームの属性のいくつかを、例外スタックの頂点における例外の属性にマッチさせることができる。双方の属性がマッチした場合には、要すれば、ホストが例外の終端を通告されて、その例外をスタックの頂点から除去することができ、したがって、スタックは、現在のトラップすなわち割込みの歴史の記録を形成することになる。事象コード＃１のスタックフレームが発生すれば、その例外スタックの頂点に新たな事象を付加することになる。
【００８７】このカートリッジを用いた非強制的な実時間デバックの例をリスト１１を参照して説明するに、タスクは典型的にはコンパクトディスクとするデータ源からＣＤ－Ｉプレーヤのメモリマップに移したデータのモジュールの検出に関するものである。ＯＳ－９モジュールは、すべてモジュル長、コードやデータ等の型式およびヘッダチェックサムを記載したものを含む多数のフィールドからなるヘッダを含めた標準書式を有している。モジュールをメモリにロードしたときは、ヘッダチェックサムを、ＯＳ－９コードによって評価するとともに、ヘッダのチェックサムフィールドに蓄積した価と比較する。これは、ヘッダが誤りなくロードされていることを手早く指示することになる。このタスクを実行するオペレーションシステムコードは、「カーネル」と呼ばれるコアＯＳ－９モジュール内にあり、このコードの知識を用いることにより、カートリッジが評価されつつあるチェックサムをウォッチし、したがってロードされつつあるモジュルを実時間で検出することになる。チェックサム評価期間中のバスアクティビティをリスト１１に示す。
【００８８】サイクル（状態）９０５および９０６は、チェックすべきヘッダのサイズをｄ１レジスタにロードする。ヘッダは２Ｅヘックスバイトの長さであり、サイクル９０６におけるシフトは、バイトアクセスよりもワードアクセスを用いてヘッダをチェックすることを考慮に入れている。サイクル９０７においては、評価したチェックサムをＦＦヘックスに初期化する。この評価ループは、サイクル９０８でスタートし、モジュールヘッダにおける次のワードをｄＯレジスタに読込む。最初のかかる読込みがサイクル９０９で認められる。サイクル９１０は、新たな値を排他的ＯＲすることにより、その新たな値を現在のチェックサムの値に加算するチェックサム評価の標準方法を用いている。サイクル９１１および９１２は、チェックサム評価が終了していない場合には、その評価ループを反復させる。
【００８９】モジュール検出用データを提供するのに用いたフィルタは、
【数３】
「moveq ＃＄２Ｅ，ｄ１」
命令に対する読取りをトリガする。アドレスＯｘ１８Ａ１ＦＣに対する読取りによりそのモジュールを検出した後に、フィルタは、引続く６４サイクルに関するデータを事象キャッシュメモリ２０（図１）中にセーブする。そのセーブしたデータには、各サイクル毎のアドレス、データおよびコントロールバスの各値が含まれ、したがって、チェックサム評価ループが生ずるのを示す。カートリッジプロセッサ１０（図１）は、このようにして、サイクル９０９における第１ヘッダフィールドアクセス

から、また長さおよび型式などヘッダ内のフィールドから、新たなモジュールがメモリ内の何処に位置するかを検出することができる。ついで、このデータは、使用者の注目を得るために、デバッギングホストＰＣに送られる。

【００９０】オペレーションシステムの知識に基づいたトリガに従い、このデータが実時間で捕捉されるので、ＣＤ－Ｉプレーヤは、遂行されるデバッグタスクに影響されずに、正常通りに機能し続ける。

【００９１】以上に説明したシステムは、バスシグナルをフィルタリングして結果の事象を解析する、という問題に対するアプローチを提供する。事象バッファの解析は、特別のシステム呼出しが検出され、タイミング情報、スタックポインタおよびシステム呼出しに対するパラメータなどの属性とともにホストデバッグシステムに報告されるようにする。

【００９２】以上に説明した本発明の開示を読めば、他の変形は当業者には自明であり、かかる変形には、モニタおよびデバッグのシステム、デバイスおよび構成要素について既知であって、以上に説明した特徴に替えて、あるいは加えて用い得る他の特徴を含めることができる。諸特徴の特別の組合わせに対する特許請求の範囲を冒頭に掲載したが、本発明の開示の範囲には、前掲の特許請求の範囲との関連の有無および本発明が解決すべき技術的問題の緩和の程度には拘わりなく、以上に明確に、あるいは暗黙理に、あるいは一般化して開示した新たな特徴もしくは特徴の新たな組合わせを含むこと勿論である。

【００９３】なお、以上に記載した本発明によるモニタカートリッジの特徴を表わすオペレーションを説明するためのコードセグメントをつぎのリスト１乃至１１に順次に示す。

【００９４】

【表１】

リスト１

```
1   Read   0x1A1F5A  0x426A   ; clr.w ...
2   Read   0x1A1F5C  0x0008   ; ... 8(a2)
3   Read   0x1A1F5E  0x4E40   ; TRAP #0 opcode

4   Write  0x0FEC28  0x0000   ; Clear taking place

5   Write  0x0FF19A  0x0080   ; Stack frame header
6   Write  0x0FF196  0x001A   ; High PC
7   Write  0x0FF198  0x1F60   ; Low PC
8   Write  0x0FF194  0x2504   ; Status word

9   Read   0x000080  0x0000   ; High vector address
10  Read   0x000082  0x062C   ; Low vector address

11  Read   0x00062C  0x4878   ; pea.l ...
12  Read   0x00062E  0x0080   ; ... (0x80).w

13  Read   0x000630  0x4E49   ; jmp ...

14  Write  0x0FF190  0x0000   ; pea.l taking place
15  Write  0x0FF192  0x777A

16  Read   0x000632  0x0018   ; ... 0x18777A
17  Read   0x000634  0x777A

18  Read   0x18777A  0x48E7   ; movem.l ...
19  Read   0x18777C  0xFFFE   ; ... d0-d7/a0-a6, -(a7)

20  Write  0x0FF18C  0x0000   ; Save a6
21  Write  0x0FF18E  0x1500

22  Write  0x0FF188  0x001F   ; Save a5
23  Write  0x0FF18A  0xFC01

24  Write  0x0FF184  0x000F   ; Save a4
25  Write  0x0FF186  0xFA94

26  Write  0x0FF180  0x001F   ; Save a3
27  Write  0x0FF182  0xFC01

28  Write  0x0FF17C  0x000F   ; Save a2
29  Write  0x0FF17E  0xEC20

30  Write  0x0FF178  0x000F   ; Save a1
31  Write  0x0FF17A  0xED05

32  Write  0x0FF174  0x001A   ; Save a0
33  Write  0x0FF176  0x1E96
```

【００９５】　　【表２】

リスト１の続き

34	Write	0x0FF170	0x0000	; Save d7
35	Write	0x0FF172	0x0000	
36	Write	0x0FF16C	0x0000	; Save d6
37	Write	0x0FF16E	0x0000	
38	Write	0x0FF168	0x0000	; Save d5
39	Write	0x0FF16A	0x2000	
40	Write	0x0FF164	0x0000	; Save d4
41	Write	0x0FF166	0x01F4	
42	Write	0x0FF160	0x0000	; Save d3
43	Write	0x0FF162	0x2C96	
44	Write	0x0FF15C	0x0000	; Save d2
45	Write	0x0FF15E	0x0001	
46	Write	0x0FF158	0x0000	; Save d1
47	Write	0x0FF15A	0x0001	
48	Write	0x0FF154	0x0000	; Save d0
49	Write	0x0FF156	0x0003	
50	Read	0x18777E	0x2C78	; movea.l ...
51	Read	0x187780	0x0000	; ... (0).w, a6
52	Read	0x000000	0x0000	; Read taking place
53	Read	0x000002	0x1500	
54	Read	0x187782	0x286E	; movea.l ...
55	Read	0x187784	0x004C	; ... 76(a6), a4
56	Read	0x00154C	0x0000	; Move taking place
57	Read	0x00154E	0x2500	
58	Read	0x187786	0x422F	; clr.b ...
59	Read	0x187788	0x0041	; ... 65(a7)
60	Read	0x18778A	0x2A6F	; movea.l ...
61	Write	0x0FF194	0x0000	; Clear taking place
62	Read	0x18778C	0x0042	; ... 66(a7), a5
63	Read	0x0FF196	0x001A	; Move taking place
64	Read	0x0FF198	0x1F60	
65	Read	0x18778E	0x3E15	; movea.w (a5), a7
66	Read	0x1A1F60	0x0084	; ISOpen function code

【0096】

【表3】

27 28

リスト2

1	Read	0x187572	0x6600	; move.w d0, (a0)+
2	Read	0xFFFFF8	0xFF80	; Fetch vector number (0x80)
3	Write	0x05D436	0x0200	; Stack frame header
4	Write	0x05D432	0x0018	; High PC
5	Write	0x05D434	0x7572	; Low PC
6	Write	0x05D430	0x2004	; Status word
7	Read	0x000200	0x0000	; High vector address
8	Read	0x000202	0x09EC	; Low vector address
9	Read	0x0009EC	0x4878	; Jump to vector's contents

リスト3

50	Read	0x18777E	0x2C78	; movea.l ...
51	Write	0x0FF152	0x00F4	; Stack frame header
52	Write	0x0FF14E	0x0018	; High PC
53	Write	0x0FF150	0x777E	; Low PC
54	Write	0x0FF14C	0x2004	; Status word
55	Read	0x0000F4	0x0000	; High vector address
56	Read	0x0000F6	0x074E	; Low vector address
57	Read	0x00074E	0x4878	; Jump to vector's contents

リスト4

1	Read	0x18762C	0x4E73	; RTE
2	Read	0x0FD4B8	0x0000	; Status word
3	Read	0x0FD4BE	0x0080	; Stack frame header
4	Read	0x0FD4BA	0x000F	; High PC
5	Read	0x0FD4BC	0x4142	; Low PC
6	Read	0x0F4142	0x2C78	; movea.l ...

【0097】

【表4】

リスト5

```
0   Read   0x1887EE  0x4E90   ; jsr (a0)
1   Read   0x191900  0x2469   ; Read of opcode
2   Write  0x0FD444  0x0018   ; Write PC high
3   Write  0x0FD446  0x87F0   ; Write PC low
4   Read   0x191902  0x0004   ; Next opcode
```

リスト6

```
0   Read   0x1885BE  0x61E2   ; bsr disp=0xE2
1   Read   0x1885A2  0x0C40   ; Read of opcode
2   Write  0x0FD45C  0x0018   ; Write PC high
3   Write  0x0FD45E  0x85C0   ; Write PC low
4   Read   0x1885A4  0x0020   ; Next opcode
```

リスト7

```
1   Write   STACK - 2     HEADER         ; LRWN6 = 0, LD6 & 0xF000 = 0
2   Write   STACK - 6     HIGH_PC        ; LRWN5 = 0
3   Write   STACK - 4     LOW_PC         ; LRWN4 = 0
4   Write   STACK - 8     STATUS         ; LRWN3 = 0
5   Read    VECTOR        HIGH_NEW_CODE  ; LRWN2 = 1, LA2 = LD6
6   Read    VECTOR + 2    LOW_NEW_CODE   ; LRWN1 = 1
7   Read    NEW_CODE      OPCODE         ; LRWN0 = 1, LA0 = (LD2 << 8) | L
```

リスト8

```
1   Read    STACK         STATUS         ; LRWN4 = 1
2   Read    STACK + 6     HEADER         ; LRWN3 = 1, LD3 & 0xF000 = 0
3   Read    STACK + 2     HIGH_NEW_CODE  ; LRWN2 = 1
4   Read    STACK + 4     LOW_NEW_CODE   ; LRWN1 = 1
5   Read    NEW_CODE      OPCODE         ; LRWN0 = 1, LA0 = (LD2 << 8) |
```

【0098】

【表5】

リスト９

```
1   Read   CODE       0x48E7      ; LRWN1 = 1, LD1 = 0x48E7
2   Read   CODE + 2   0xFFFE      ; LRWN0 = 1, LD0 = 0xFFFE
```

リスト１０

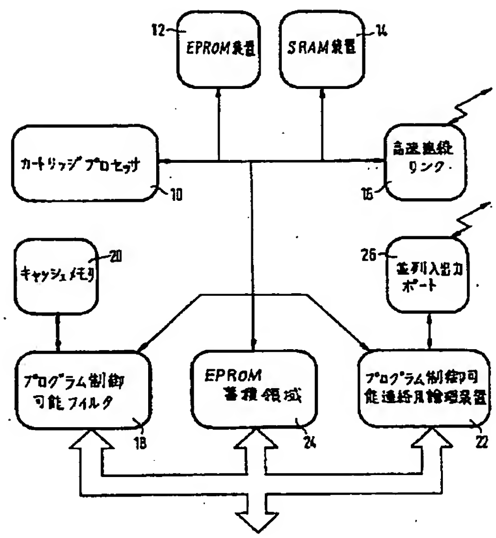
```
1   Read   CODE       0x3E15      ; LRWN1 = 1, LD1 = 0x3E15
2   Read   DATA       FUNCTION    ; LRWN0 = 1
```

リスト１１

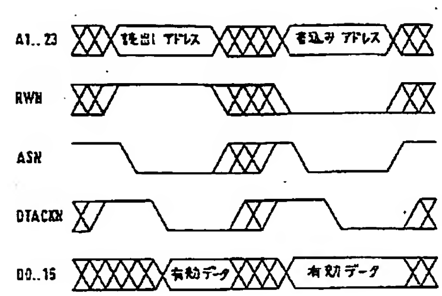
```
CYCLE -MODULE- -ADDR- DATA ACCESS -------INTERPRETATION-------
00905 kernel   18A1FC 722E  Read  moveq    #$2E,d1

00906 kernel   18A1FE E249  Read  lsr.w    #1,d1

00907 kernel   18A200 74FF  Read  moveq    #$FF,d2

00908 kernel   18A202 3018  Read  move.w   (a0)+,d0

00909          92CBC0 4AFC  Read  move.w   manifest cycle

00910 kernel   18A204 B142  Read  eor.w    d0,d2

00911 kernel   18A206 51C9  Read  dbf      d1,...
00912 kernel   18A208 FFFA  Read           ...SFFFA
00913 kernel   18A202 3018  Read  move.w   (a0)+,d0

00914          92CBC2 0001  Read  move.w   manifest cycle

00915 kernel   18A204 B142  Read  eor.w    d0,d2

00916 kernel   18A206 51C9  Read  dbf      d1,...
00917 kernel   18A208 FFFA  Read           ...SFFFA
```

【図面の簡単な説明】
【図１】本発明を実施するコンパクトディスクプレーヤ用モニタカートリッジを示すブロック線図である。
【図２】読出し・書込み周期のタイミングを示す線図である。
【図３】図１に示したプログラム制御可能論理ユニットの他の構成例を示すブロック線図である。

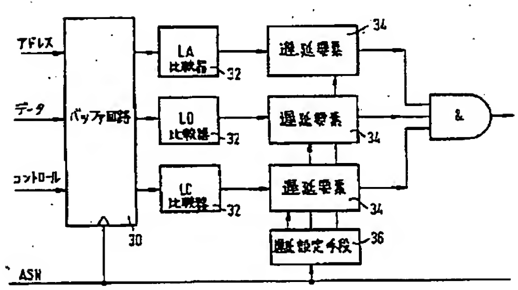
【図１】



【図２】



【図３】



フロントページの続き

(72)発明者　ポウル　アンドリュー　クラーク
イギリス国　アールエイチ11　７ディーエイチ　ウエスト　サセックス　クロウレイ　クロウレイ　ホスピタル　ザ　ガレイジフラット（番地なし）

(72)発明者　ジョナサン　リチャード　ビーシング
イギリス国　シーアール０　７エイチエイ　クロイドン　ノーサンプトン　ロード20